
5.1 Segment Translation

Figure 5-2 shows in more detail how the processor converts a logical address into a linear address.

To perform this translation, the processor uses the following data structures:

- Descriptors
- Descriptor tables
- Selectors
- Segment Registers

5.1.1 Descriptors

The segment descriptor provides the processor with the data it needs to map a logical address into a linear address. Descriptors are created by compilers, linkers, loaders, or the operating system, not by applications programmers. Figure 5-3 illustrates the two general descriptor formats. All types of segment descriptors take one of these formats. Segment-descriptor fields are: gigabyte = 2 ^ 30 bit

BASE: Defines the location of the segment within the 4 gigabyte linear address space. The processor concatenates the three fragments of the base address to form a single 32-bit value.

LIMIT: Defines the size of the segment. When the processor concatenates the two parts of the limit field, a 20-bit value results. The processor interprets the limit field in one of two ways, depending on the setting of the granularity

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

bit:

megabyte = 2 ^ 20 bit

1. In units of one byte, to define a limit of up to 1 megabyte. kilobyte = 2 ^ 10 bit
2. In units of 4 Kilobytes, to define a limit of up to 4 gigabytes. The limit is shifted left by 12 bits when loaded, and low-order one-bits are inserted.

Granularity bit: Specifies the units with which the LIMIT field is interpreted. When thebit is clear, the limit is interpreted in units of one byte; when set, the limit is interpreted in units of 4 Kilobytes.

TYPE: Distinguishes between various kinds of descriptors.

DPL (Descriptor Privilege Level): Used by the protection mechanism (refer to Chapter 6) .

Segment-Present bit: If this bit is zero, the descriptor is not valid for use in address transformation; the processor will signal an exception when a selector for the descriptor is loaded into a segment register. Figure 5-4 shows the format of a descriptor when the present-bit is zero. The operating system is free to use the locations marked AVAILABLE. Operating systems that implement segment-based virtual memory clear the present bit in either of these cases:

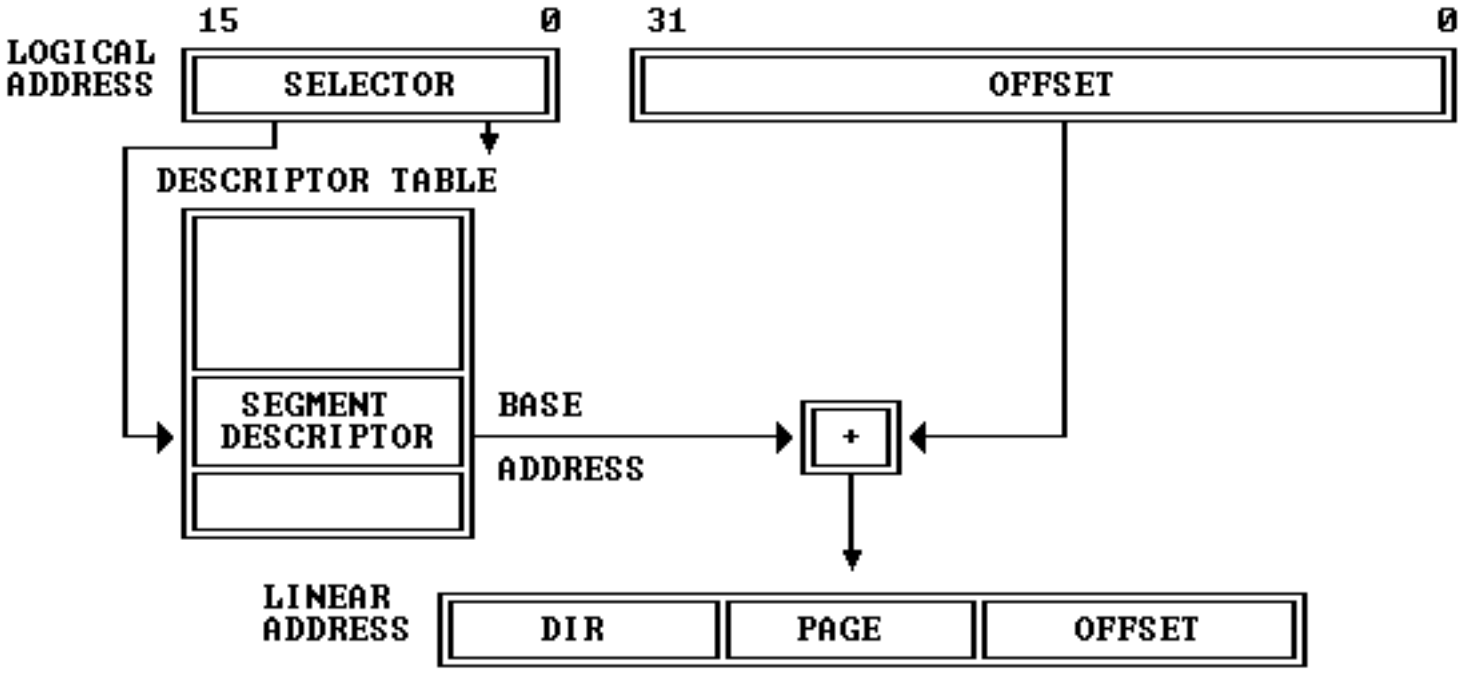
- When the linear space spanned by the segment is not mapped by the paging mechanism.
- When the segment is not present in memory.

Accessed bit: The processor sets this bit when the segment is accessed; i.e., a selector for the descriptor is loaded into a segment register or used by a selector test instruction. Operating systems that implement virtual memory at the segment level may, by periodically testing and clearing this bit, monitor frequency of segment usage.

Creation and maintenance of descriptors is the responsibility of systems software, usually requiring the cooperation of compilers, program loaders or system builders, and therating system.

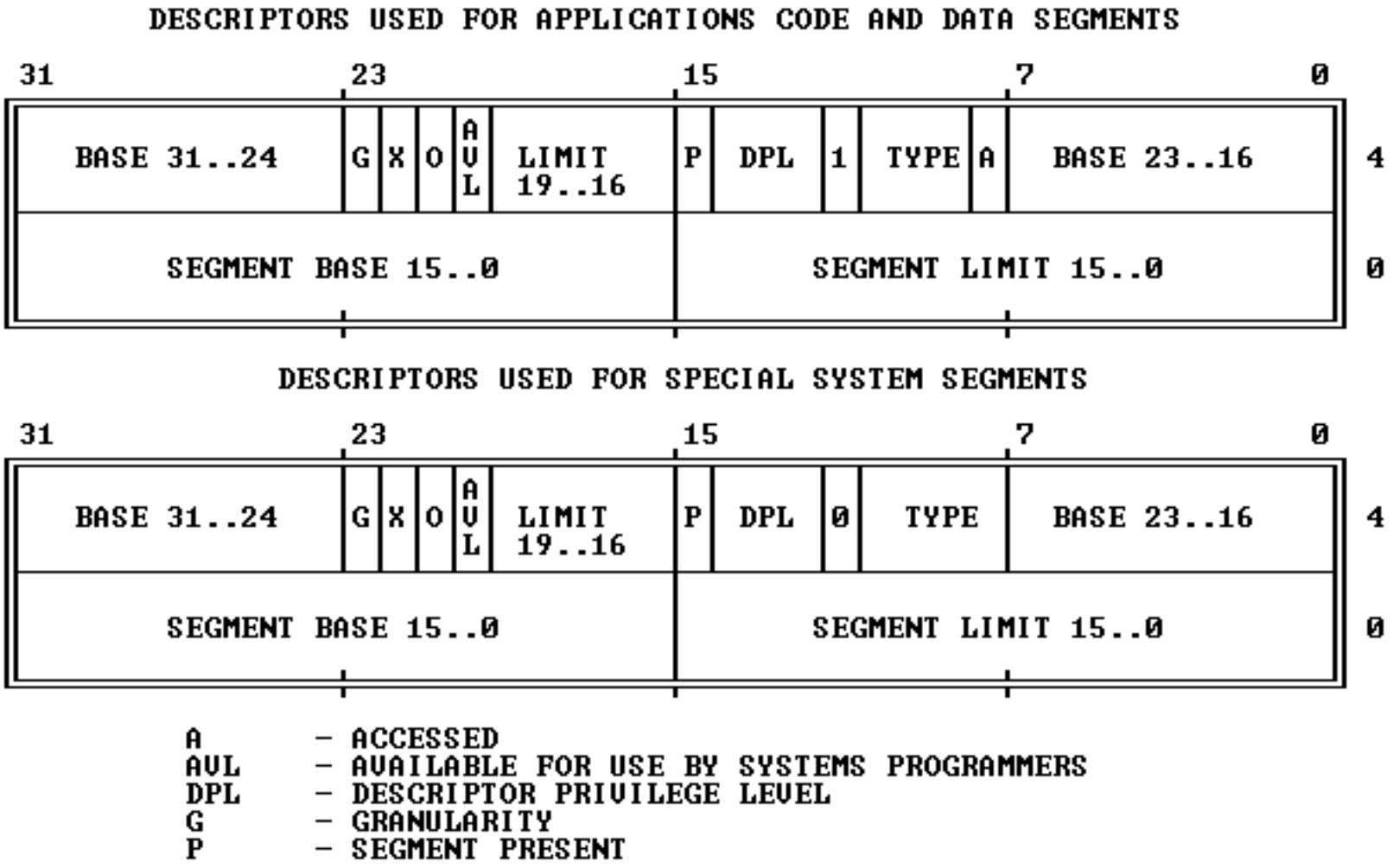
open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Figure 5-2. Segment Translation



open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Figure 5-3. General Segment-Descriptor Format



5.1.2 Descriptor Tables

Segment descriptors are stored in either of two kinds of descriptor table:

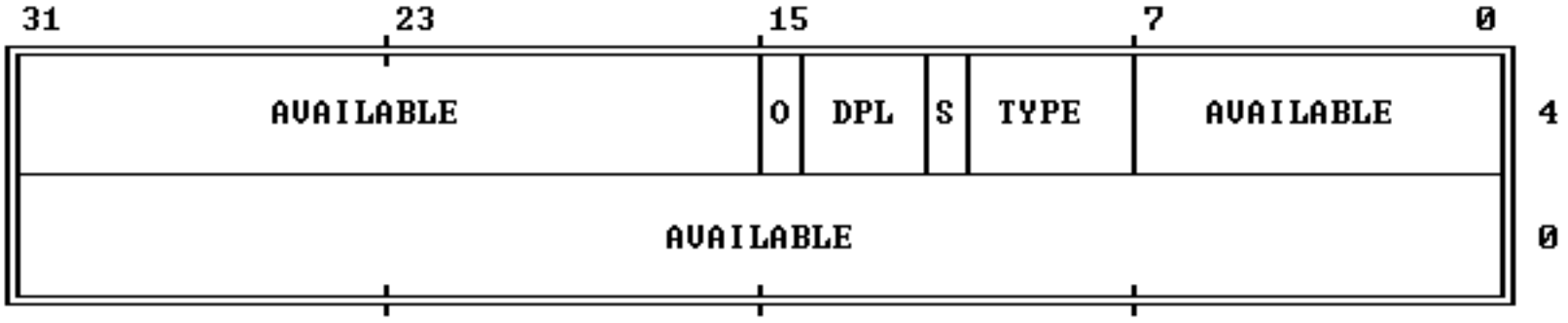
- The global descriptor table (GDT)
- A local descriptor table (LDT)

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

A descriptor table is simply a memory array of 8-byte entries that contain descriptors, as Figure 5-5 shows. A descriptor table is variable in length and may contain up to 8192 (2^(13)) descriptors. The first entry of the GDT (INDEX=0) is not used by the processor, however.

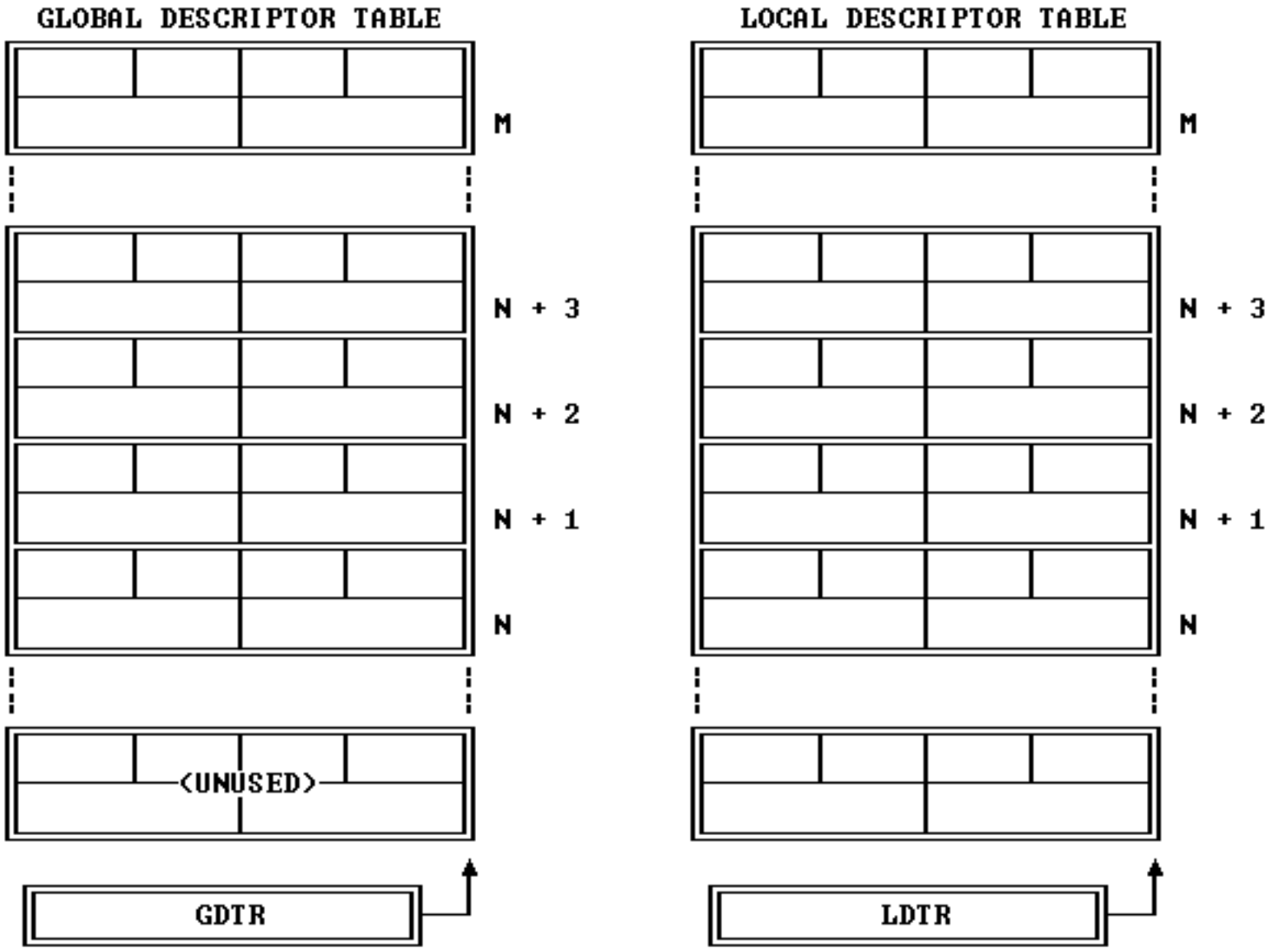
The processor locates the GDT and the current LDT in memory by means of the GDTR and LDTR registers. These registers store the base addresses of the tables in the linear address space and store the segment limits. The instructions LGDT and SGDT give access to the GDTR; the instructions LLDT and SLDT give access to the LDTR.

Figure 5-4. Format of Not-Present Descriptor



open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Figure 5-5. Descriptor Tables



5.1.3 Selectors

The selector portion of a logical address identifies a descriptor by specifying a descriptor table and indexing a

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

descriptor within that table. Selectors may be visible to applications programs as a field within a pointer variable, but the values of selectors are usually assigned (fixed up) by linkers or linking loaders. Figure 5-6 shows the format of a selector.

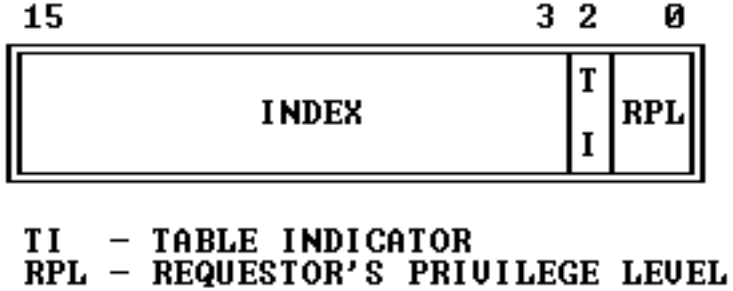
Index: Selects one of 8192 descriptors in a descriptor table. The processor simply multiplies this index value by 8 (the length of a descriptor), and adds the result to the base address of the descriptor table in order to access the appropriate segment descriptor in the table.

Table Indicator: Specifies to which descriptor table the selector refers. A zero indicates the GDT; a one indicates the current LDT.

Requested Privilege Level: Used by the protection mechanism. (Refer to Chapter 6)

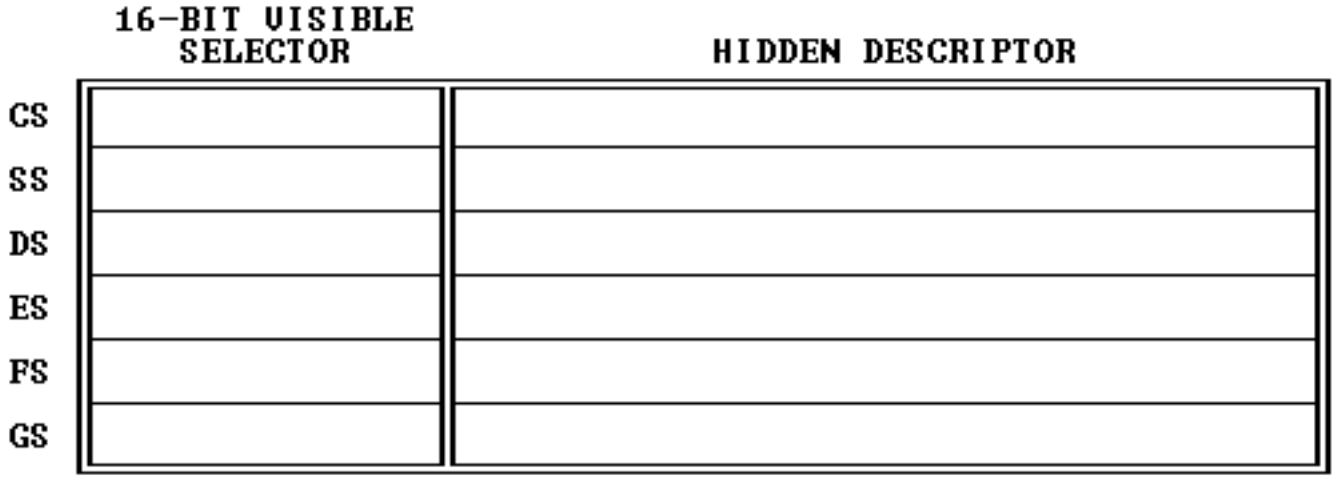
Because the first entry of the GDT is not used by the processor, a selector that has an index of zero and a table indicator of zero (i.e., a selector that points to the first entry of the GDT), can be used as a null selector. The processor does not cause an exception when a segment register (other than CS or SS) is loaded with a null selector. It will, however, cause an exception when the segment register is used to access memory. This feature is useful for initializing unused segment registers so as to trap accidental references.

Figure 5-6. Format of a Selector



open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Figure 5-7. Segment Registers



5.1.4 Segment Registers

The 80386 stores information from descriptors in segment registers, thereby avoiding the need to consult a descriptor table every time it accesses memory.

Every segment register has a "visible" portion and an "invisible" portion, as Figure 5-7 illustrates. The visible portions of these segment address registers are manipulated by programs as if they were simply 16-bit registers. The invisible portions are manipulated by the processor.

The operations that load these registers are normal program instructions (previously described in Chapter 3). These instructions are of two classes:

1. Direct load instructions; for example, MOV, POP, LDS, LSS, LGS, LFS. These instructions explicitly reference the segment registers.
2. Implied load instructions; for example, far CALL and JMP. These instructions implicitly reference the CS register, and load it with a new value.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Using these instructions, a program loads the visible part of the segment register with a 16-bit selector. The processor automatically fetches the base address, limit, type, and other information from a descriptor table and loads them into the invisible part of the segment register.

Because most instructions refer to data in segments whose selectors have already been loaded into segment registers, the processor can add the segment-relative offset supplied by the instruction to the segment base address with no additional overhead.

---

up: Chapter 5 -- Memory Management
prev: Chapter 5 -- Memory Management
next: 5.2 Page Translation

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com